Front End 15 Hz Operation

Linac characterization

Fri, Feb 23, 2007

Introduction

With increasing demands placed upon the Fermilab accelerator for delivery of more beam, the Booster will be required to deliver beam at rates approaching 15 Hz. The Linac has always been able to deliver beam at 15 Hz. This note describes characteristics of the design of its front end control system software that make it convenient to support 15 Hz operation.

15 Hz world

The Linac accelerator is driven by a timing system that results in beam accelerated at 15 Hz. The timing is derived from the Booster magnet circuit and is roughly synchronized to the 60 Hz power line frequency. Although the power line frequency varies slightly during the day, the Booster and Linac track that variation. In order to monitor the 15 Hz Linac, the front ends operate in synchronism with the Linac. About 1 ms after the 50 μs Linac beam pulse, the front ends receive an interrupt that is based upon a clock event plus delay. The interrupt starts a series of tasks that all run to completion in a portion of the 66 ms cycle.

The first job is to refresh the data pool, so that the readings of all devices are stored into memory. The series of steps required to make this happen is determined by the contents of the Data Access Table, whose entries are interpreted in sequence to update the data pool. One of the last of these entries specifies that all local applications are to be called to perform any work needed, often referencing the contents of the data pool as it exists at that moment. After the data pool has been updated, all active data requests for which replies are due on the current cycle are fulfilled and delivered to the requesting nodes, again, often referencing the fresh data pool readings. After that, an alarm scan is made covering all devices not bypassed, and any resulting alarm messages are queued to the network. Finally, the "little console" active page application is called to run.

The time required for all of this work is a small portion of the 15 Hz cycle. The remainder of the cycle is mostly idle, except for network activity and clock event interrupts. Front end activity repeats in this way every 15 Hz cycle. The front end "breathes" with the accelerator.

A data request can arrive via the network at any time, of course, and a one-shot request will receive a prompt reply. If the data request is for data in the data pool, it is merely sampled from the current data pool, and the reply message is delivered. There is no need to do anything special to acquire the data from hardware, as the data already resides in the data pool. The Data Access Table updates the data pool in the same way every cycle. All data is available.

The data pool is always consistent. Even when a request arrives at a time early in the cycle when the data pool is being updated, processing of that request is held back until all the data pool is refreshed. There is no way for an external requester to "see" a partially updated data pool.

Closed loops

A closed loop is implemented as a local application, which is a function that can be compiled and downloaded into a front end even during normal operation. The connection with the ongoing system operation is merely via the calls that the system makes to this function every 15 Hz cycle. The timing of the call is when the data pool has just been updated, so that any data needed by the closed loop is fresh. If the closed loop requires making an adjustment in a device, it can do it right then, still early in the cycle.

Memory file system

Each front end includes a nonvolatile memory card that houses system tables as well as a

file system used mainly for application code. System code is downloaded via the network at boot time, but the applications are taken from the local file system as needed. System tables hold the current readings, settings, and alarm parameters for all analog devices, plus the current state of all digital devices. When a node boots, which may occur after a power outage, all device settings are delivered to the hardware. The setting values are known because they are maintained in the nonvolatile memory tables; there is no need to obtain them first from another source.

Local applications are stored in nonvolatile memory. A special nonvolatile memory table, called `LATBL`, houses a set of parameters for each LA, or more accurately, for each LA instance of an LA. To run a closed loop on multiple devices of the same type, multiple entries in `LATBL` can lead the system to call the same piece of code multiple times, each time with a separate parameter set. The first time an LA instance call is made, the application allocates a context memory block and places a pointer to that block in the parameter list, so that on subsequent calls, the code is reminded of all it needs to know for operating that LA instance. Normally, an application runs forever, but if it is desired to disable an LA instance, its enable bit parameter can be set to zero. The system will then make a final call, and the LA will "clean up its act" by releasing any system resources it allocated for use during its operation. Note that when a system boots, the automatic settings restore includes restoring the enable bits, one for each LA instance. That means that all LAs that were active the last time the system was running are automatically brought back into operation. Once an LA instance has been installed, it effectively becomes a permanent part of that front end's operation.

Correlated data

As an aid to collecting data from multiple front ends that can be relied upon to stem from the same 15 Hz cycle, a data request can be given server-style support. When a data request is received from an Acnet requester that asks for a number of devices not all coming from that node, the receiving front end accepts the responsibility of collecting the required data from the various front ends and delivering it in a single composite reply to the original requesting node. The node does this by using multicast to forward the request message to all the other front ends in the Linac. Each of the Linac front ends, then, scans the request to see whether it can contribute to fulfilling the request. If it can, it initializes the request, and each time a reply is due, it sends its contribution to the "server" node. The server node accepts all these anticipated replies and delivers a complete reply message to the original requester. Every such reply message is correlated; all data was measured on the same 15 Hz cycle. The synchronous operation of all the front ends makes this collection of correlated data achievable.

Little consoles

Many Linac front ends use a little console that permits convenient local access for Linac technicians. Most often, this means calling up a list of readings on a parameter page that update at about 1 Hz. There are many page applications available, too, although they mostly find use in configuring and diagnosing front end operation. An important aspect to the support for all page applications is that access to data from other front ends is transparent. For example, it is easy to set up a list of parameters that display the RF gradient reading of the first 5 Linac accelerating cavities that is viewable from the little console of any node. The readings displayed are all measured on the same 15 Hz cycle. Actually, the readings are usually averages, such that the readings of all beam cycles are averaged; in the absence of any beam cycles, one gets the straight average.

The page applications that run on a little console can even be used when there is no little console hardware connected to the front end. This can be done from any host platform that runs a "Page G" emulation of little console displays. In this case, the page application runs in the target front end node, but it is operated from the host display.